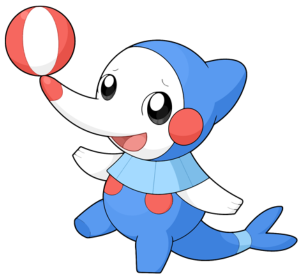
数值计算之美

SHU ZHI JI SUAN ZHI MEI

胡家威

http://hujiaweibujidao.github.io/



清华大学逸夫图书馆 · 北京

内容简介

本书是我对数值计算中的若干常见的重要算法及其应用的总结，内容还算比较完整。

本人才疏学浅，再加上时间和精力有限，所以本书不会详细介绍很多的概念，需要读者有一定的基础或者有其他的参考书籍，这里推荐参考文献中的几本关于数值计算的教材。

本书只会简单介绍下算法的原理，对于每个算法都会附上我阅读过的较好的参考资料以及算法的实现 (Matlab 或者其他语言)，大部分代码是来源于参考文献 [1] 或者是经过我改编而成的，肯定都是可以直接使用的，需要注意的是由于 Latex 对代码的排版问题，导致中文注释中的英文字符经常出现错位，对于这种情况请读者自行分析，不便之处还望谅解。写下这些内容的目的是让自己理解地更加深刻些，顺便能够作为自己的 HandBook，如有错误之处，还请您指正，本人邮箱地址是：hujiawei090807@gmail.com。

目　　录

第二章　非线性方程求解

非线性方程 $f(x)=0$ 的解法有很多，本章主要介绍的方法有二分法、牛顿法、割线法、逆二次插值法、Zeroin 算法等等。

2.1　二分法

二分法很简单，每次计算区间中点处的函数值，判断它和区间右端点正负号的关系，然后将区间缩小一半进一步逼近方程的解。算法很简单，直接附上可以使用的源码，需要注意的是循环退出的条件一定要合适！

code/bisectionnm.m

```
function x = bisectionnm(F,a,b)
% 二分法求非线性方程的根
% F=@(x) x-sqrt(2);
% bisectionnm(F,1,2) %1.414213562373095
while (b-a) > eps*abs(b)
    x=(a+b)/2;
    if (sign(F(x))==sign(F(a)))
        a=x;
    else
        b=x;
    end
end
end
```

2.2 牛顿法

牛顿法是在函数 $f(x)$ 上的当前点 x_n 处画一条切线，切线与 x 轴的交点，就是对方程解的下一个逼近值 x_{n+1}，然后一直重复迭代下去直到满足阈值条件退出。其中 x_{n+1} 和 x_n 满足如下条件：

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

牛顿法的优点是非常高效，减少了很多的迭代步骤，它是二次收敛的 (具体说明参考文献 [1](P106))。但是它有严重的缺陷：(1) 要求函数 $f(x)$ 必须是光滑的;(2) 可能遇到 $f'(x)$ 不好计算的情况；(3) 初始解要接近准确解。如果 $f(x)$ 不具有连续的、有界的一阶、二阶导数，或者初始解没有足够接近准确解时，牛顿法可能收敛得很慢，或者根本不收敛。比较典型的例子就是下面的函数，用牛顿法求解时会出现迭代解一直往返于 $x = a$ 的两侧。

$$f(x) = sign(x-a)\sqrt{|x-a|}$$

牛顿法的 Matlab 实现代码：

code/newtonnm.m

```
function x = newtonnm(F,FP,x)
% 牛顿求非线性方程的根
% F=@(x) x^2-2;
% FP=@(x) 2*x;
% newtonnm(F,FP,1) %1.414213562373095

xprev=0;
while abs(x-xprev) > eps*abs(x)
    xprev=x;
    x=x-F(x)/FP(x);
end
end
```

2.3 割线法

割线法使用最近两次迭代解构造出的有限差分近似，代替牛顿法中的求导数计算，它通过两个点画一条割线，割线与 x 轴的交点就是下一个迭代解，所以割线法需要两个初始的迭代值，迭代公式如下：

$$x_{n+1} = x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$$

割线法是超线性收敛的，优点是不需要显示计算 $f'(x)$，却有和牛顿法接近的收敛速度。使用割线法可以解决上节中牛顿法不能收敛的函数 $f(x) = sign(x-a)\sqrt{|x-a|}$。下面是割线法的 Matlab 实现代码：

code/secantnm.m

```
function b = secantnm(F,a,b)
% 割线法求非线性方程的根
% F=@(x) x^2-2;
% secantnm(F,1,2) %1.414213562373095

while abs(b-a) > eps*abs(b)
    c=a;%保存上上次解
    a=b;%保存上次解
    b=b+(b-c)/(F(c)/F(b)-1);%计算这次的解
end
end
```

2.4　逆二次插值法

逆二次插值法 (Inverse Quadratic interpolation=IQI) 使用三个近似解来产生下一个迭代解。假设已知了三个值，a、b 和 c，以及对应的函数值，$f(a)$、$f(b)$ 和 $f(c)$，可以用一个抛物线来对这三个点进行插值，然后令抛物线与 x 轴的交点是下一个迭代解，但是问题是这种情况下抛物线与 x 轴不一定有交点！所以，可以换成考虑关于 y 的二次函数，它是一个侧向抛物线 $P(y)$，插值条件是：

$$a = P(f(a)) \quad b = P(f(b)) \quad c = P(f(c))$$

这个抛物线一定与 x 轴有交点，在交点处 $y = 0$，对应的 $x = P(0)$ 为下一步迭代解。IQI 算法的问题是，多项式插值要求数据点的横坐标 (此处即为 $f(a)$、$f(b)$ 和 $f(c)$) 互不相同，但这是无法保证的。下面是 IQI 法的 Matlab 代码实现：

code/iqinm.m

```
function x = iqinm(F,a,b,c)
% 逆二次插值法求非线性方程的根
% F=@(x) sign(x-2)*sqrt(abs(x-2));
% iqinm(F,1,3,4) %2

while abs(c-b) > eps*abs(c)
    %多项式插值，因为长度是，所以阶数是32
    x=polyinterp([F(a),F(b),F(c)],[a,b,c],0);
    a=b;
    b=c;
    c=x;
end
end
```

2.5 Zeroin 算法

Zeroin 算法的核心思想就是将二分法的可靠性和割线法与 IQI 法的收敛速度结合起来。算法步骤：

(1) 选取初始值 a 和 b，使得 $f(a)$ 和 $f(b)$ 的正负号正好相反；

(2) 使用一步割线法，得到 a 和 b 之间的一个值 c；

(3) 重复下面的步骤，直到 $|b-a|<\epsilon|b|$ 或者 $f(b)=0$；

(4) 重新排列 a、b 和 c (可能需要经过两轮)，使得：

-$f(a)$ 和 $f(b)$ 的正负号相反；

-$|f(b)|\leq|f(a)|$；

-c 的值为上一步 b 的值。

(5) 如果 $c\neq a$，执行 IQI 算法中的一步迭代；

(6) 如果 $c=a$，执行割线法中的一步迭代；

(7) 如果执行一步 IQI 算法或者割线法得到的近似解在区间 $[a,b]$ 内，接受这个解为 c；

(8) 如果这个解不在区间 $[a,b]$ 内，执行一步二分法得到 c。

从算法迭代过程可以看出：b 是当前最优解，$|f(b)|$ 最接近 0，c 是次最优解 (也就是 b 的上一次值)，$[b,a]$ 是有根区间。算法按照“IQI 法、割线法、二分法”的优先顺序来计算下一步迭代解，保证较快的收敛速度，每次迭代都使得有根区间缩小，并且会抛弃“不满意”的 IQI 或者割线法得到的解 (例如解脱离了有根区间或者没有使得有根区间缩小等情况)。下面给出参看文献 [1] 中对 Matlab 中 $fzero$ 方法的精简版本 $fzerotx$ 代码，需要注意的是循环退出的条件以及接受迭代解的判断条件，比较复杂，我没理解，可以自己去看源码实现：

code/fzerotx.m

```
function b = fzerotx(F,ab,varargin)
%FZEROTX  Textbook version of FZERO.
%   x = fzerotx(F,[a,b]) tries to find a zero of F(x) between a and
    b.
%   F(a) and F(b) must have opposite signs.  fzerotx returns one
%   end point of a small subinterval of [a,b] where F changes sign.
%   Arguments beyond the first two, fzerotx(F,[a,b],p1,p2,...),
%   are passed on, F(x,p1,p2,..).
%
%   Examples:
%       fzerotx(@sin,[1,4])
%       MATLAB6: F = inline('sin(x)'); fzerotx(F,[1,4])
%       MATLAB7: F = @(x) sin(x); fzerotx(F,[1,4])

% Initialize.
a = ab(1);
b = ab(2);
fa = feval(F,a,varargin{:});
fb = feval(F,b,varargin{:});
if sign(fa) == sign(fb)
   error('Function must change sign on the interval')
end
c = a;
fc = fa;
d = b - c;
e = d;

% Main loop, exit from middle of the loop
while fb ~= 0

   % The three current points, a, b, and c, satisfy:
   %    f(x) changes sign between a and b.
   %    abs(f(b)) <= abs(f(a)).
```

```
%    c = previous b, so c might = a.
% The next point is chosen from
%    Bisection point, (a+b)/2.
%    Secant point determined by b and c.
%    Inverse quadratic interpolation point determined
%    by a, b, and c if they are distinct.

 if sign(fa) == sign(fb)
    a = c;   fa = fc;
    d = b - c;   e = d;
end
 if abs(fa) < abs(fb)
    c = b;    b = a;    a = c;
    fc = fb;  fb = fa;  fa = fc;
end

% Convergence test and possible exit
m = 0.5*(a - b);
 tol = 2.0*eps*max(abs(b),1.0);
 if (abs(m) <= tol) | (fb == 0.0)
    break
end

% Choose bisection or interpolation
 if (abs(e) < tol) | (abs(fc) <= abs(fb))
    % Bisection
    d = m;
    e = m;
 else
    % Interpolation
    s = fb/fc;
    if (a == c)
       % Linear interpolation (secant)
       p = 2.0*m*s;
```

```
            q = 1.0 - s;
        else
            % Inverse quadratic interpolation
            q = fc/fa;
            r = fb/fa;
            p = s*(2.0*m*q*(q - r) - (b - c)*(r - 1.0));
            q = (q - 1.0)*(r - 1.0)*(s - 1.0);
        end;
        if p > 0, q = -q; else p = -p; end;
        % Is interpolated point acceptable
        if (2.0*p < 3.0*m*q - abs(tol*q)) & (p < abs(0.5*e*q))
            e = d;
            d = p/q;
        else
            d = m;
            e = m;
        end;
    end

    % Next point
    c = b;
    fc = fb;
    if abs(d) > tol
        b = b + d;
    else
        b = b - sign(b-a)*tol;
    end
    fb = feval(F,b,varargin{:});
end
```

参考文献

[1] Numerical Computing with Matlab. Cleve B. Moler.
中文翻译版本《Matlab 数值计算》，喻文健，机械工业出版社，2006，6
网站资源：
(1)Cleve Moler 撰写的教科书
(2)数值计算交互演示网站

[2] 数值分析与算法，喻文健，清华大学出版社，2012，1

[3] Numerical Methods: An introduction with Applications Using Matlab. Amos Gilat. Vish Subramaniam，2010，10

[4] Data Mining Algorithms In R.
网站资源:
WikiBook：Data Mining Algorithms In R